Twinder A Search Engine for Twitter Streams是Delft工学院Ke Tao在2012年ICWE上的论文。论文介绍了搜索引擎的架构以及查询的十个基础假设。



核心部件是：

1. Feature Extraction

负责解析twitter的消息并对结果进行索引，分为两类：topic-insensitive和topic-sensitive。

1. Feature Extraction Task Broker

对特征抽取任务进行管理，将其分派到cloud computing上。

1. Relevance Estimation

计算tweets之间的相关性，从而获得更准确的结果。（论文还没有发布，ppt中没有详细介绍，不知道如何计算相关性的）

另外十个假设用来计算每个tweet的权重，这些假设是根据架构中的5个特征所得。如

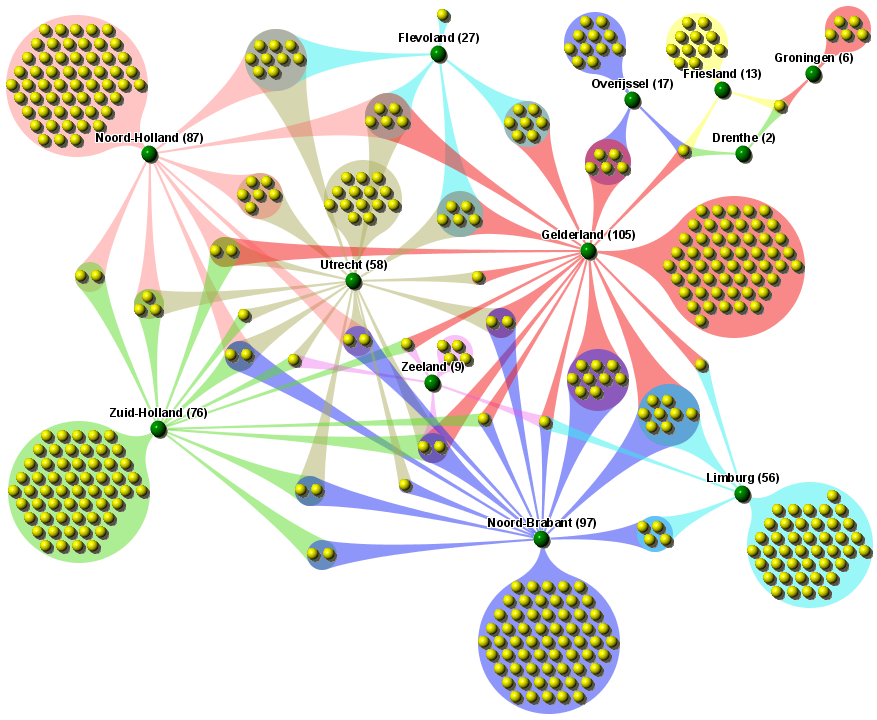
Hypothesis 4: tweets that contain hashtags are more likely to be relevant than tweets that do not contain hashtags.



从论文的整体来看目的是一样的，都是为了搜索twitter得到尽可能多和准确的结果。但是不同地方在于：

1. twinder并不关注twitter当中人的分类关系
2. twinder并不注重用户搜索关键字相关主题（topic extension）对搜索结果的影响
3. twinder仍然是基于传统的搜索方式，以文本方式显示，没有可视化的查询结果。

我所设想的搜索可以描述为：根据关键字搜索，所搜到结果应该主要是人。人以group组织，group之间通过主题词连接，主题词中包含与其语义相近的词。



主要过程包括：

1. 解析文档。找到句子中关键词，分析关键词之间的关系。
2. 构造语义图模型。按照一定的结构存储分析后的结果。
3. 子图搜索：根据给定的查询语句，从存储数据中寻找相关记录。
4. 可视化：根据查询关键字及其扩展（相似语义单词），将查询所得的用户以图表示。

其中最困难的阶段就是第一和第二个。解析句子通常有两种方式：根据词袋（Bag of Words）分析文档的主题，从而对文档进行分类，如LDA、LSA；根据本体(Ontology)（手工标注而得）分析句子，将文本连接到某个本体关键词上。

第二种方式包含了大量人工干预过程（主要是annotation），例如工具GATE或MMAX2，都需要根据文档的上下文指定一些分类并对文本进行标注后，才能将文本生成RDF模型组织的数据。当然这样处理的优点在于文档分析准确，歧义单词减少，实体之间的关系保持了文档中的原来状态。

相对来说，第一种自然语言处理的过程似乎更加可取。NLTK是python语言下处理自然语言的工具，它包含如下几个过程：segmentation, normalization, named entity recognition. 同时也提供classification, stemming, tagging, parsing, and semantic reasoning。下一步需要看的是NLTK自带的一本自然语言处理的书。